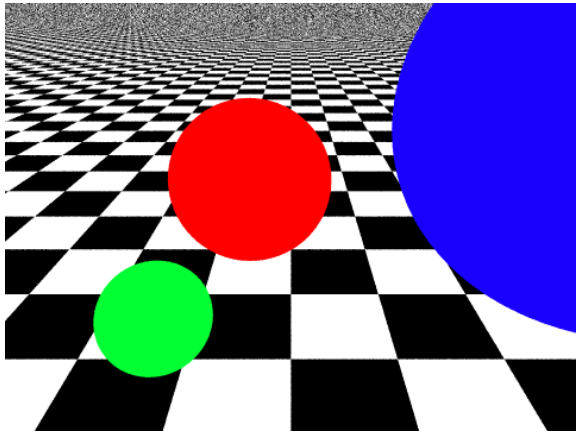
Computer Graphics

Lighting - Global

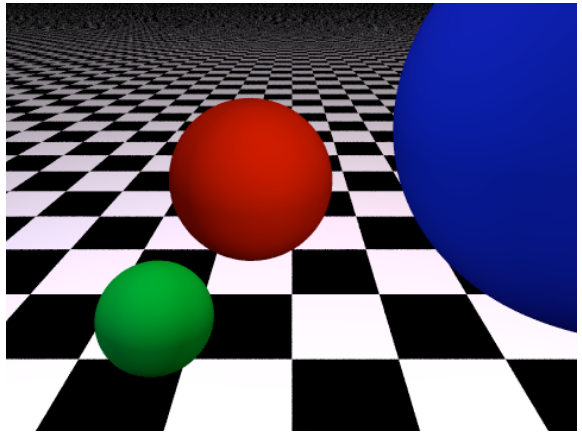
Mark Pauly

Geometric Computing Laboratory

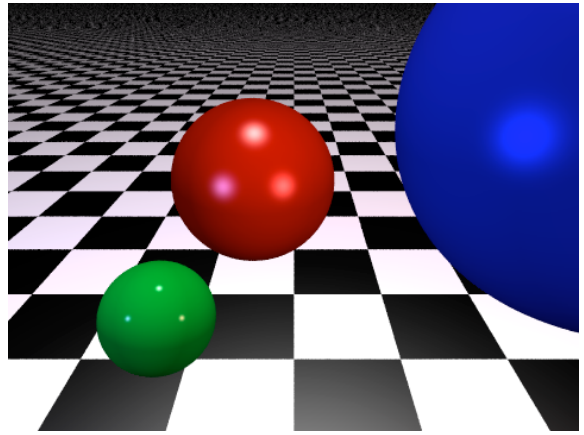
Lighting Computations



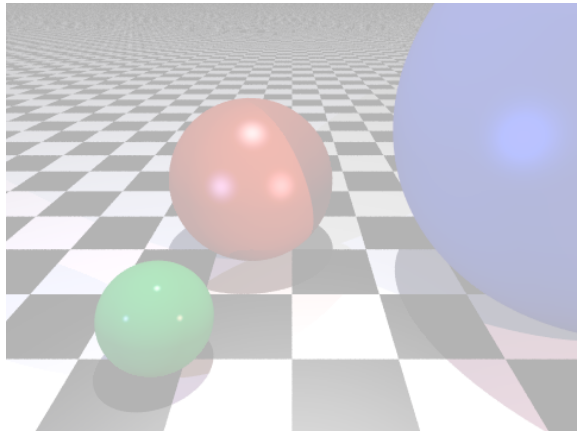
ambient



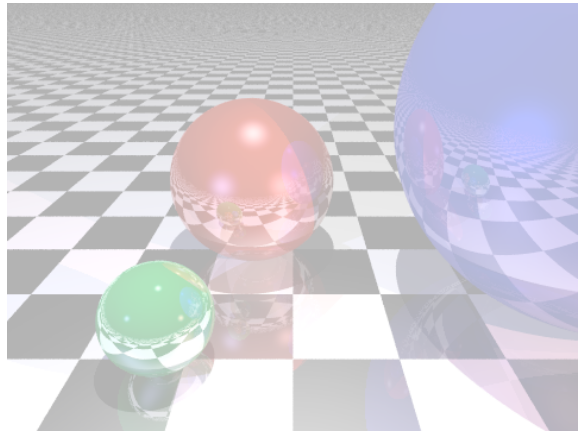
+diffuse



+specular



+shadows

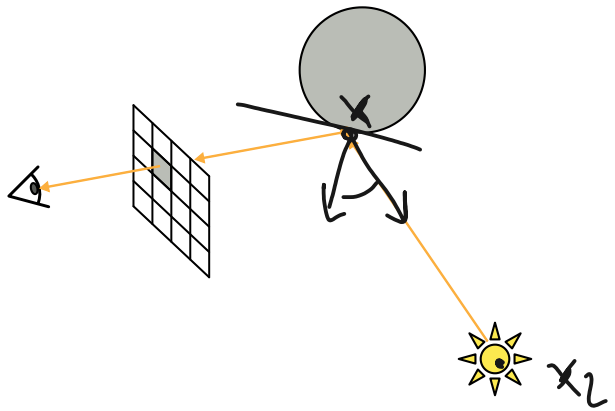


+reflections

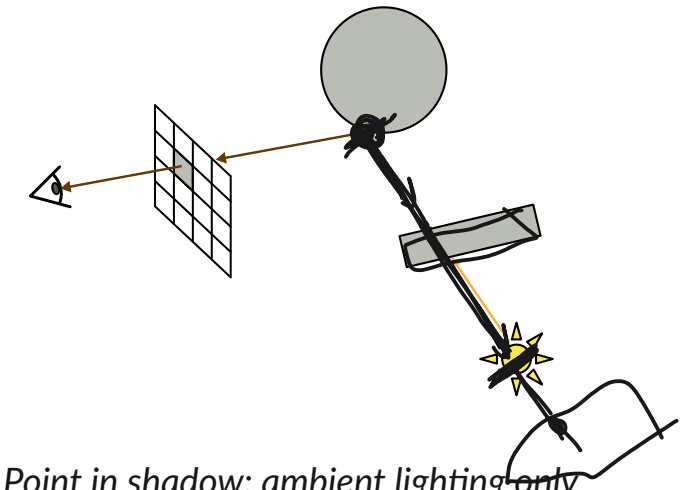
Shadows

- Send *shadow ray* from intersection point $\mathbf{x}$ to light source position $\mathbf{x}_l$.
  - Set $\mathbf{shadow}(\mathbf{x}, \mathbf{x}_l)$ to 0 if an occluding object is found and to 1 otherwise.
- Discard diffuse and specular contribution if light is blocked by another object.

$$I = I_a m_a + \sum_l I_l \cdot \mathbf{shadow}(\mathbf{x}, \mathbf{x}_l) \cdot \left(m_d \left(\mathbf{n} \cdot \mathbf{l}_l\right) + m_s (\mathbf{r}_l \cdot \mathbf{v})^s\right)$$



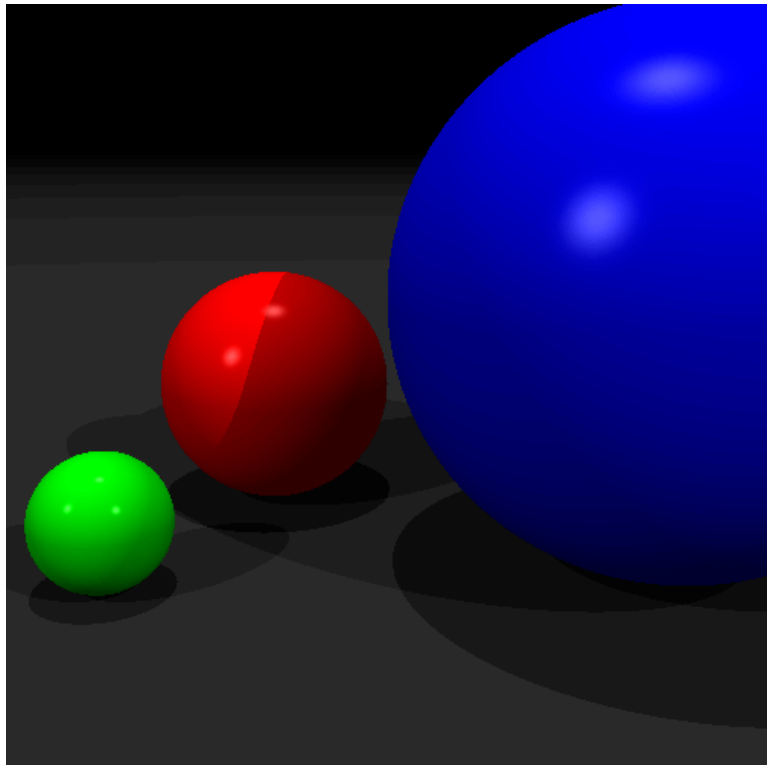
Point in light: ambient + diffuse + specular



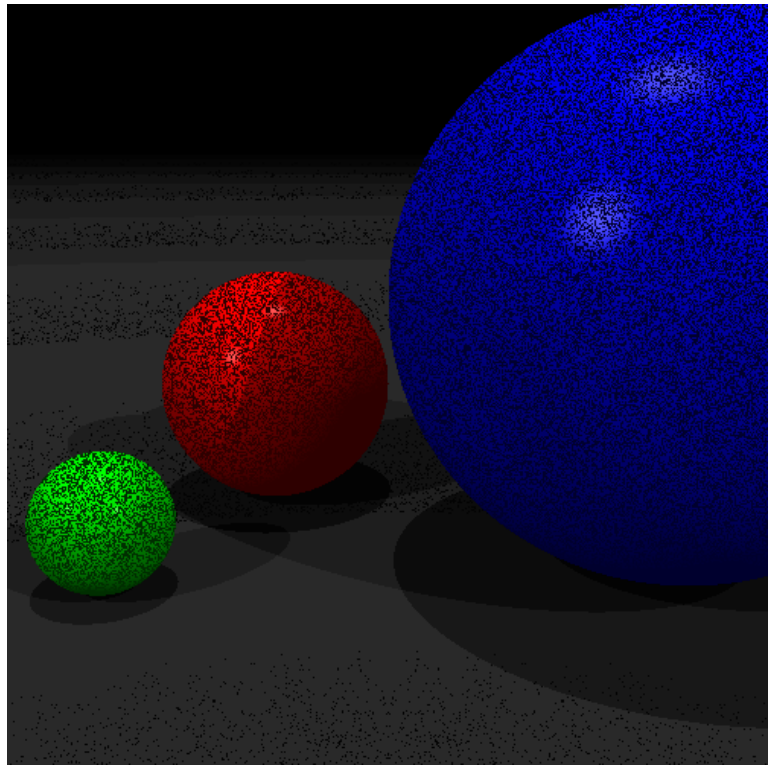
Point in shadow: ambient lighting only

Shadows

- Send *shadow ray* from intersection point $\mathbf{x}$ to light source position $\mathbf{x}_l$.
  - Set $\mathbf{shadow}(\mathbf{x}, \mathbf{x}_l)$ to 0 if an occluding object is found and to 1 otherwise.
- Discard diffuse and specular contribution if light is blocked by another object.



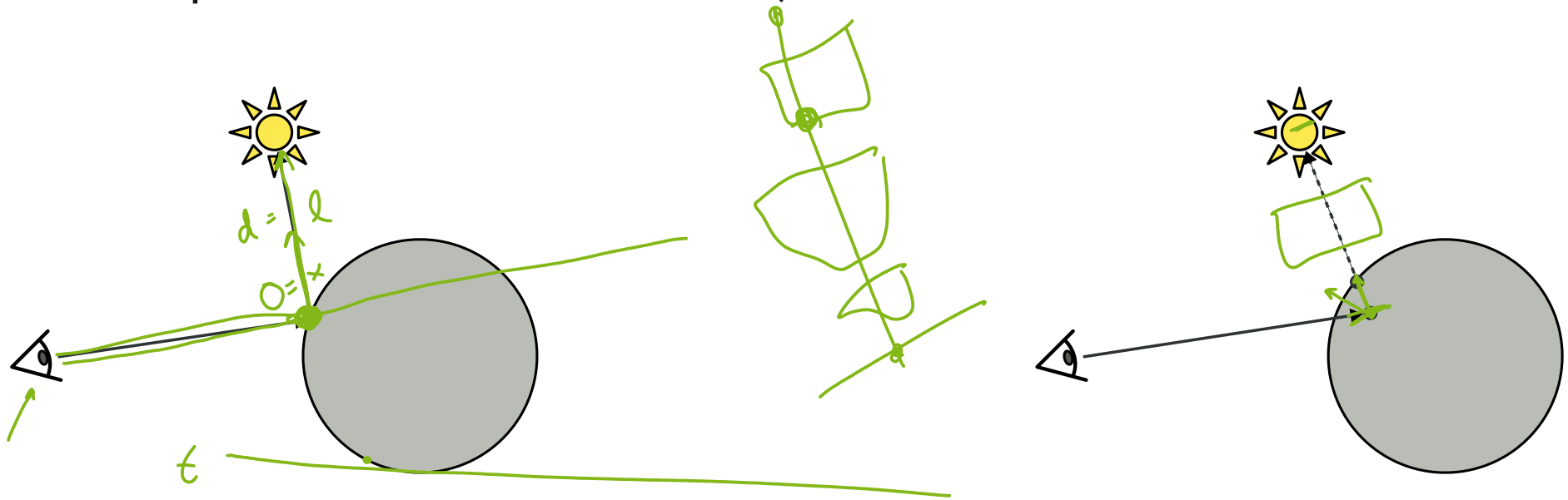
we want this



but we get this

Shadows

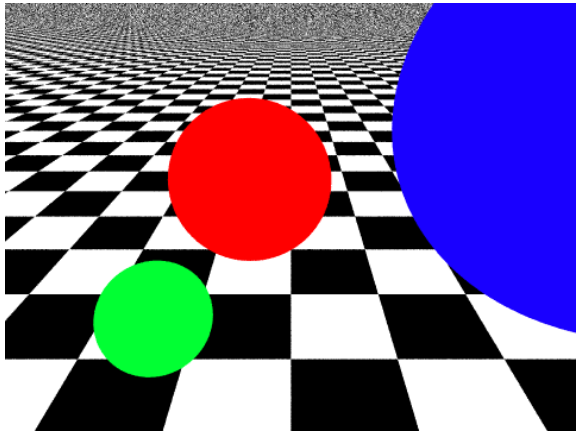
- Floating point errors might lead to erroneous self-shadowing (*shadow acne*).
- Discard intersection points of the shadow ray that are too close to the ray origin (the point $\mathbf{x}$ on the sphere that we want to test).



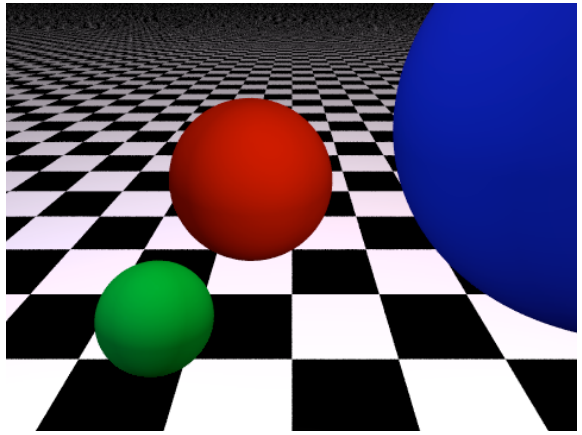
exact computation

precision problems

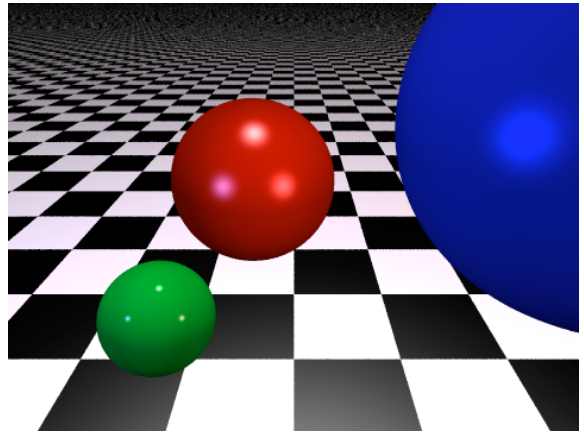
Lighting Computations



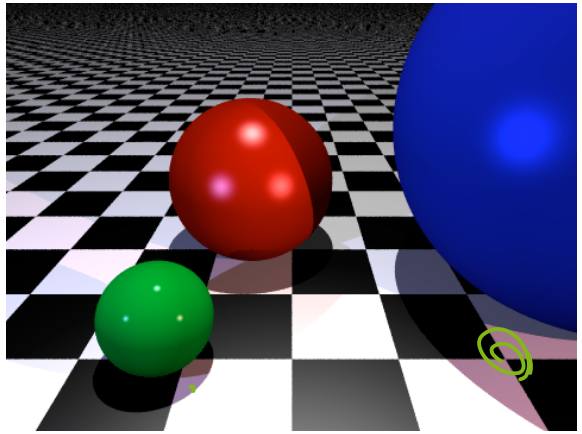
ambient



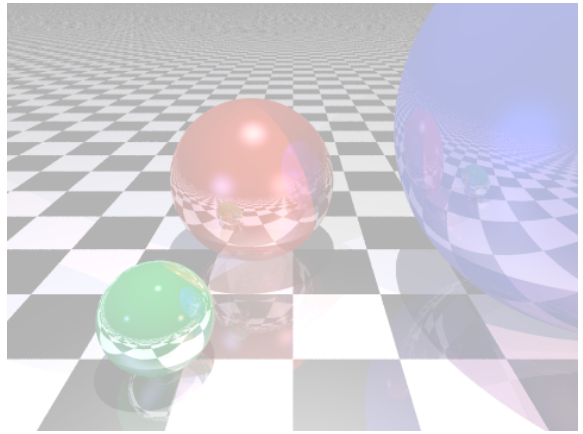
+diffuse



+specular

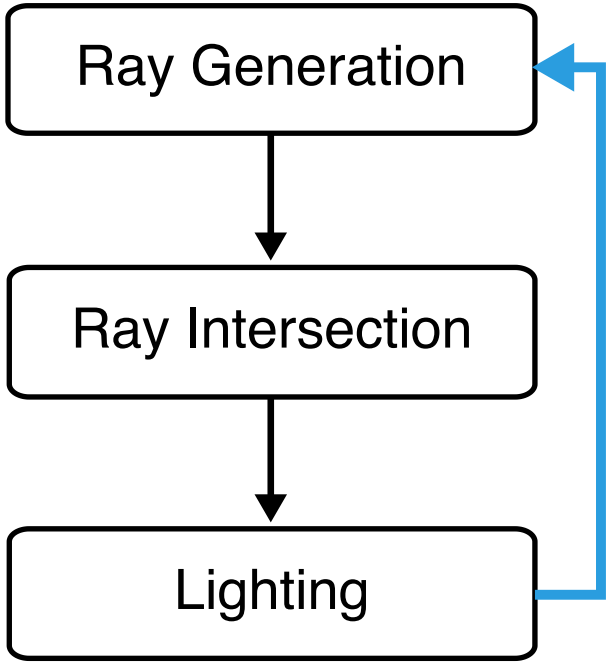


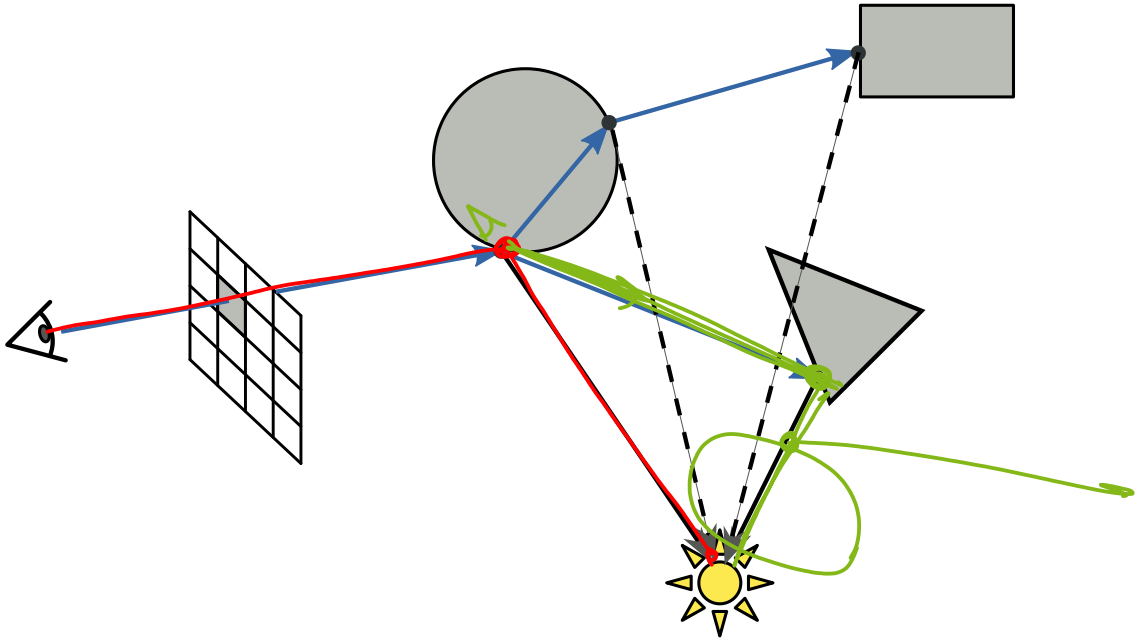
+shadows



+reflections

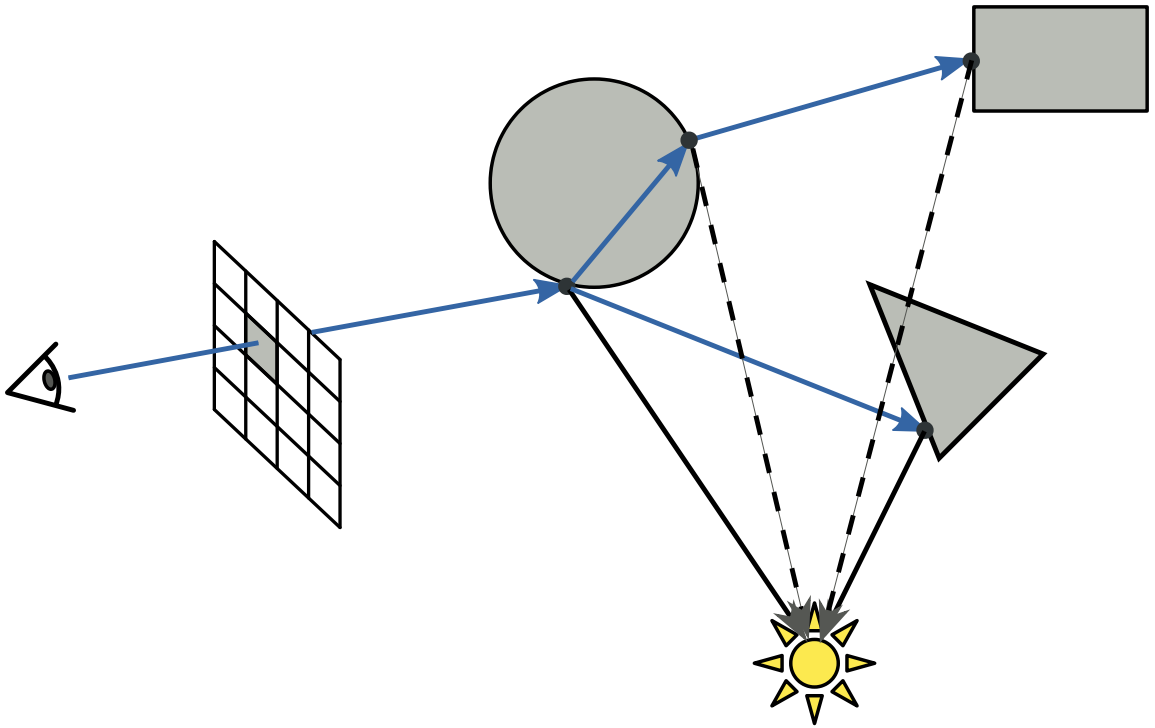
Ray Tracing Operators





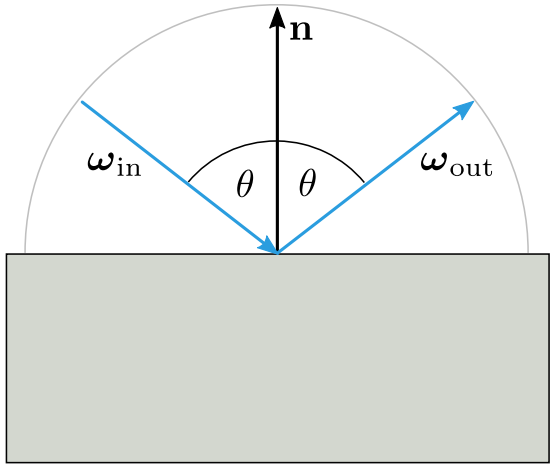
Recursive Ray Tracing

- Phong lighting models *local* illumination. Now we add *global* illumination effects.
- At each intersection point, we reflect and/or refract the incoming viewing ray at the surface normal and trace the child rays *recursively*.

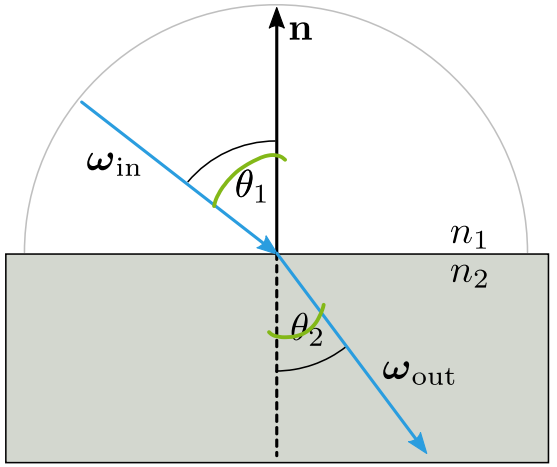


Recursive Ray Tracing

- Phong lighting models *local* illumination. Now we add *global* illumination effects.
- At each intersection point, we reflect and/or refract the incoming viewing ray at the surface normal and trace the child rays *recursively*.



$$\omega_{out} = \left(\mathbf{I} - 2\mathbf{n}\mathbf{n}^{\mathsf{T}}\right)\omega_{in}$$

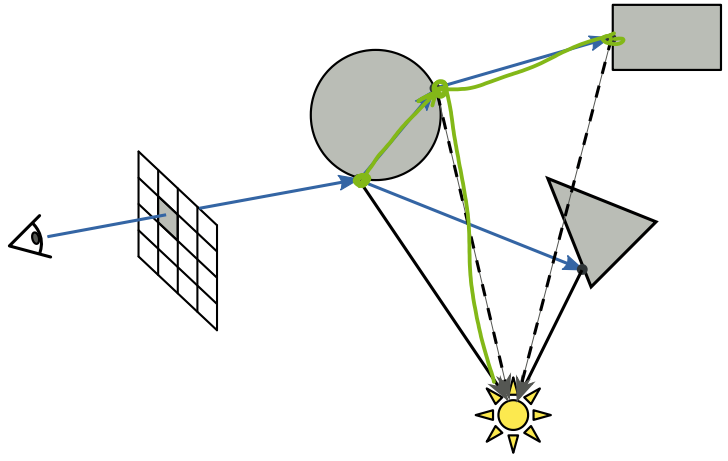


$$n_1 \sin\theta_1 = n_2 \sin\theta_2$$

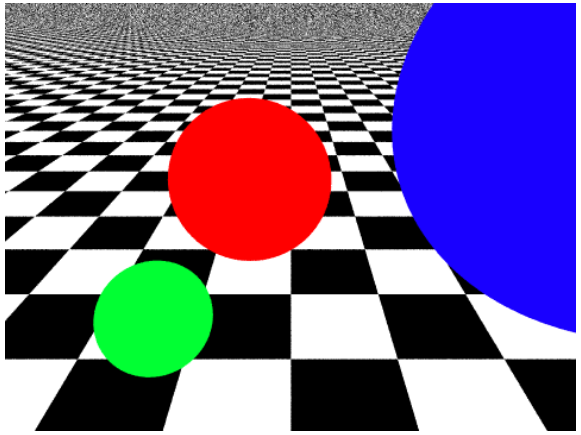
Snell's law with refraction indices n_1, n_2.

Recursive Ray Tracing

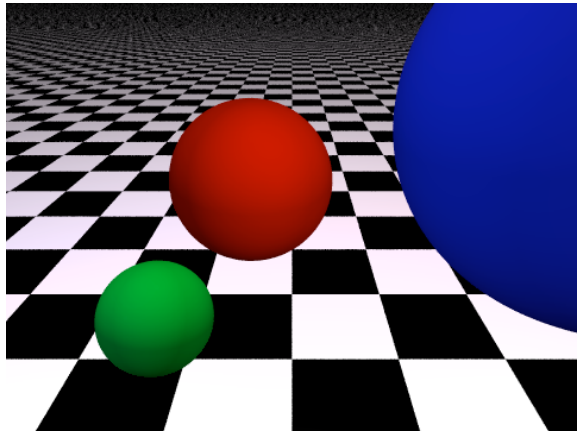
- Phong lighting models *local* illumination. Now we add *global* illumination effects.
- At each intersection point, we reflect and/or refract the incoming viewing ray at the surface normal and trace the child rays *recursively*.
- The final color is interpolated (mixed) between local Phong illumination, reflection, and refraction. The weighting of these contributions depends on material properties.



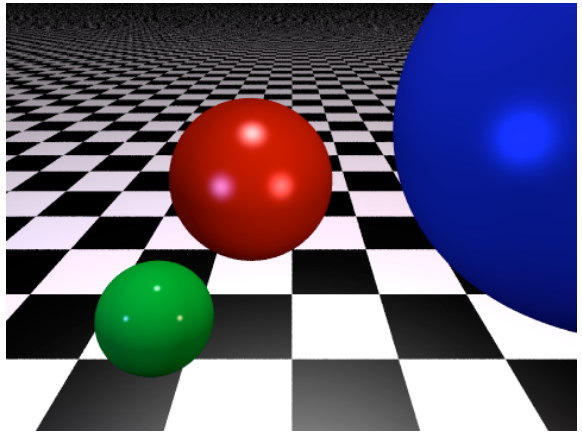
Lighting Computations



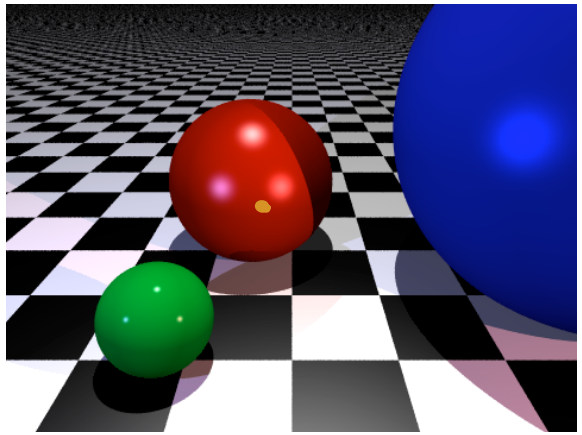
ambient



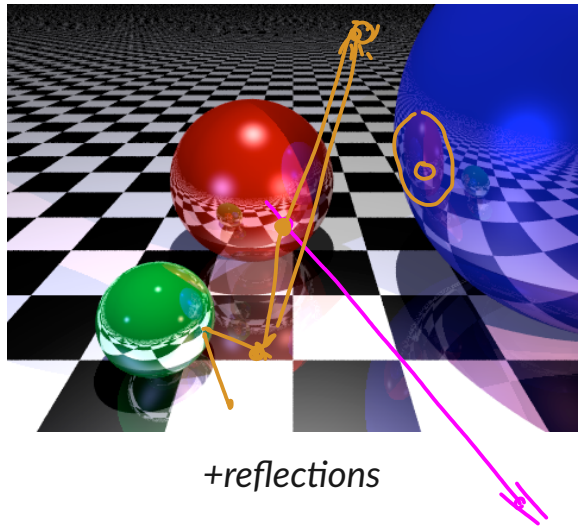
+diffuse



+specular



+shadows



+reflections

Quiz: Diffuse Lighting

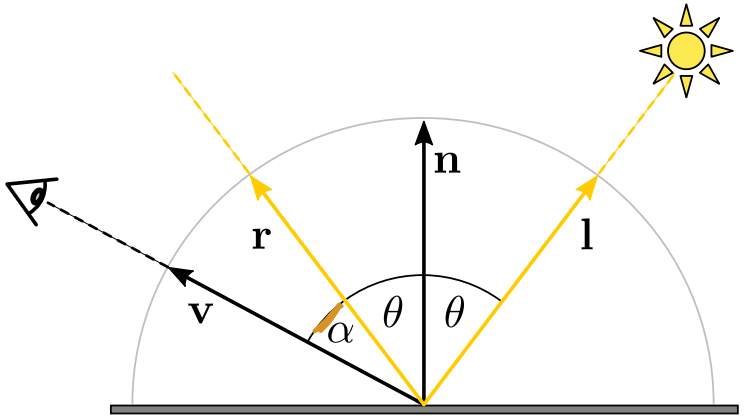
What is the diffuse component of Phong lighting?

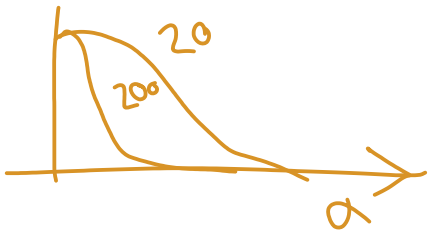
A: $I_l k_d(\mathbf{r} \cdot \mathbf{v})$

B: $I_l k_d(\mathbf{n} \times \mathbf{l})$

C: $I_l k_d(\mathbf{n} \cdot \mathbf{l})$

D: $I_l k_d(\mathbf{r} \times \mathbf{v})$

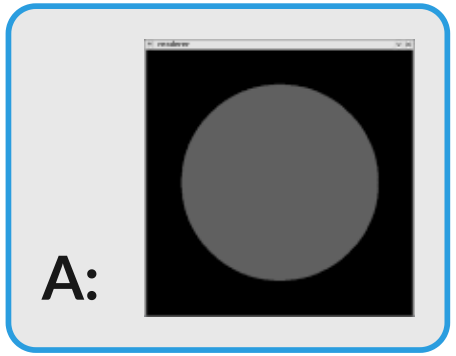


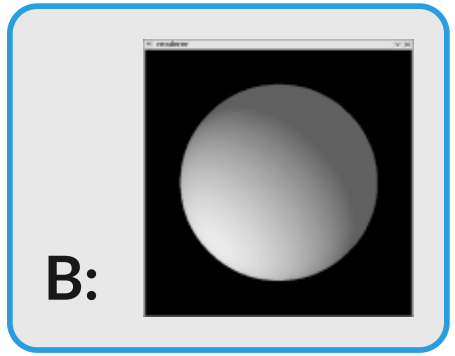


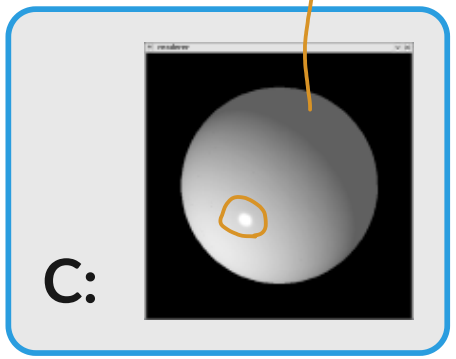
Quiz: Specular Lighting

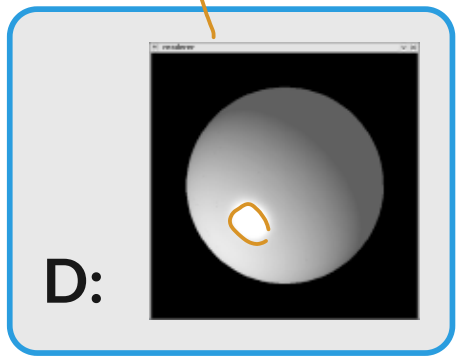
Which image matches the material listed in the bottom row?

ambient	diffuse	specular	shininess
0.3	0.0	0	0
0.3	0.6	0	0
0.3	0.6	1	20
0.3	**0.6**	**1**	**200**



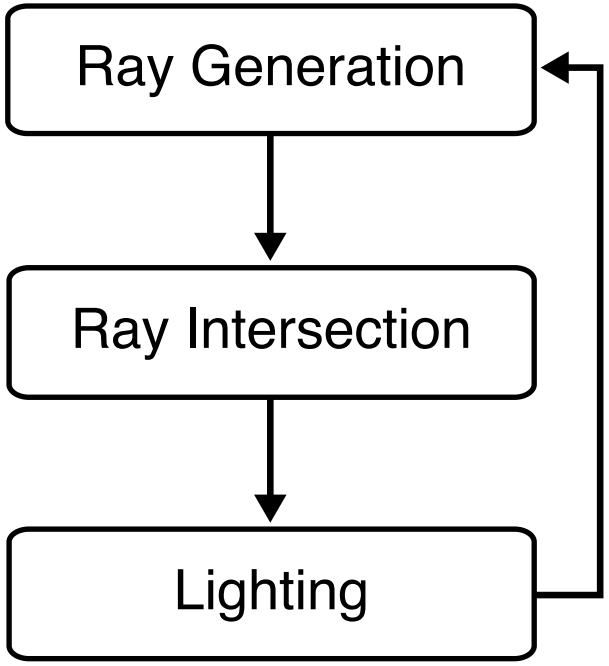


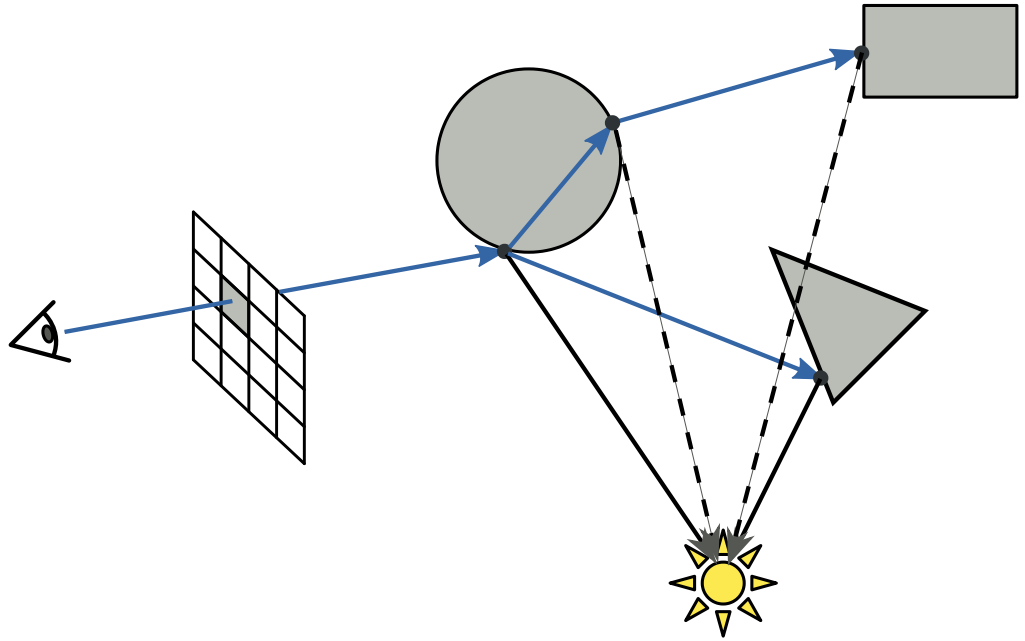




Ray Tracing Pipeline

- Now you know about ray generation, ray intersection, lighting computations, and recursive ray tracing.
- That's all you need to implement your first ray tracer!





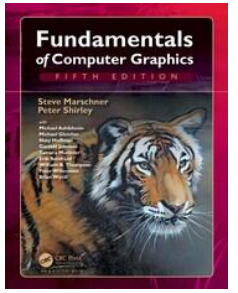
Is it difficult to code?

```
typedef struct{double x,y,z}vec;vec U,black,amb={.02,.02,.02};struct sphere{
vec cen,color;double rad,kd,ks,kt,kl,ir}*s,*best,sph[]={0.,6.,.5,1.,1.,1.,.9,
.05,.2,.85,0.,1.7,-1.,8.,-.5,1.,.5,.2,1.,.7,.3,0.,.05,1.2,1.,8.,-.5,.1,.8,.8,
1.,.3,.7,0.,0.,1.2,3.,-6.,15.,1.,.8,1.,7.,0.,0.,0.,.6,1.5,-3.,-3.,12.,.8,1.,
1.,5.,0.,0.,0.,.5,1.5,};yx;double u,b,tmin,sqrt(),tan();double vdot(A,B)vec A
,B;{return A.x*B.x+A.y*B.y+A.z*B.z;}vec vcomb(a,A,B)double a;vec A,B;{B.x+=a*
A.x;B.y+=a*A.y;B.z+=a*A.z;return B;}vec vunit(A)vec A;{return vcomb(1./sqrt(
vdot(A,A)),A,black);}struct sphere*intersect(P,D)vec P,D;{best=0;tmin=1e30;s=
sph+5;while(s-->sph)b=vdot(D,U=vcomb(-1.,P,s->cen)),u=b*b-vdot(U,U)+s->rad*s
->rad,u=u>0?sqrt(u):1e31,u=b-u>1e-7?b-u:b+u,tmin=u>=1e-7&&u<tmin?best=s,u:
tmin;return best;}vec trace(level,P,D)vec P,D;{double d,eta,e;vec N,color;
struct sphere*s,*l;if(!level--)return black;if(s=intersect(P,D));else return
amb;color=amb;eta=s->ir;d= -vdot(D,N=vunit(vcomb(-1.,P=vcomb(tmin,D,P),s->cen
)));if(d<0)N=vcomb(-1.,N,black),eta=1/eta,d= -d;l=sph+5;while(l-->sph)if((e=l
->kl*vdot(N,U=vunit(vcomb(-1.,P,l->cen))))>0&&intersect(P,U)==l)color=vcomb(e
,l->color,color);U=s->color;color.x*=U.x;color.y*=U.y;color.z*=U.z;e=1-eta*
eta*(1-d*d);return vcomb(s->kt,e>0?trace(level,P,vcomb(eta,D,vcomb(eta*d-sqrt
(e),N,black))):black,vcomb(s->ks,trace(level,P,vcomb(2*d,N,D)),vcomb(s->kd,
color,vcomb(s->kl,U,black))));}main(){printf("%d %d\n",32,32);while(yx<32*32)
U.x=yx%32-32/2,U.z=32/2-yx++/32,U.y=32/2/tan(25/114.5915590261),U=vcomb(255.,
trace(3,black,vunit(U)),black),printf("%.0f %.0f %.0f\n",U);}/*minray!*/
```

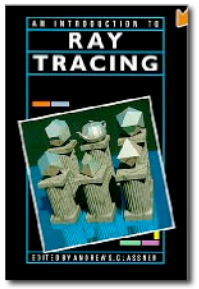
Paul Heckbert's Minimal Ray Tracer

Literature

- Marschner & Shirley: *Fundamentals of Computer Graphics*, 5th Edition, AK Peters, 2021.
  - Chapter 4

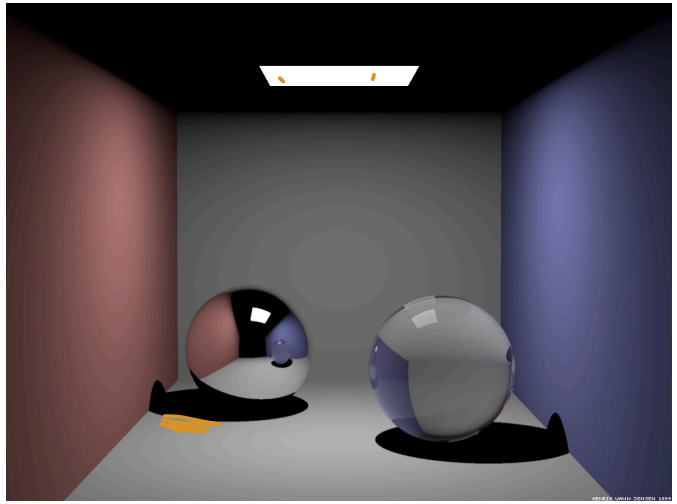


- Glassner: An Introduction to Ray Tracing, Academic Press, 1989.
  - Chapters 2, 7

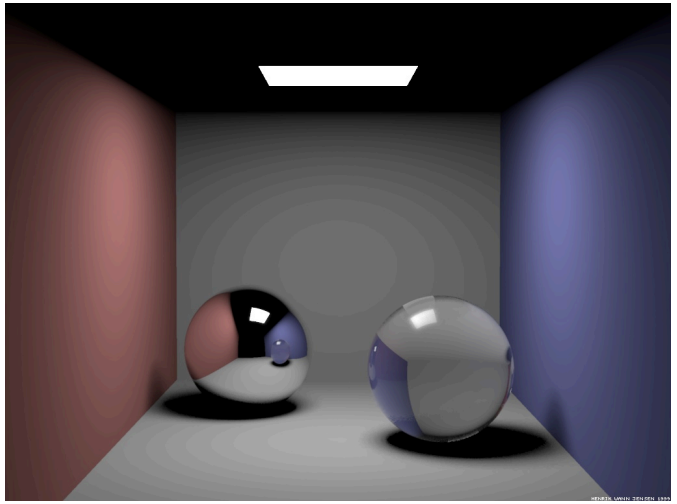


Photorealistic Rendering

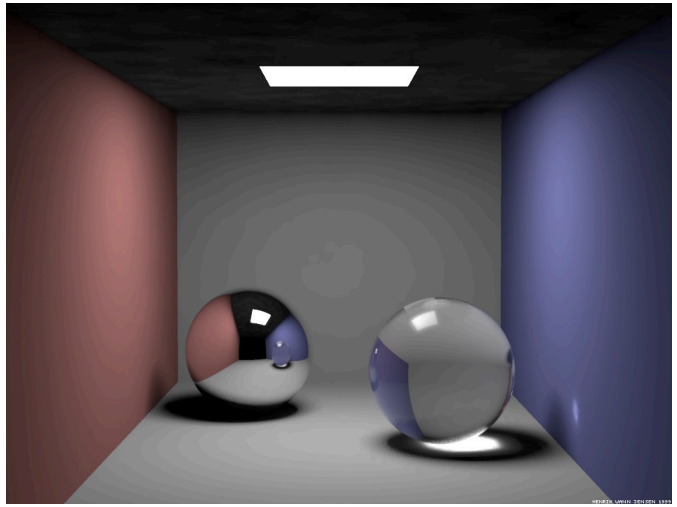
The Quest for Realism



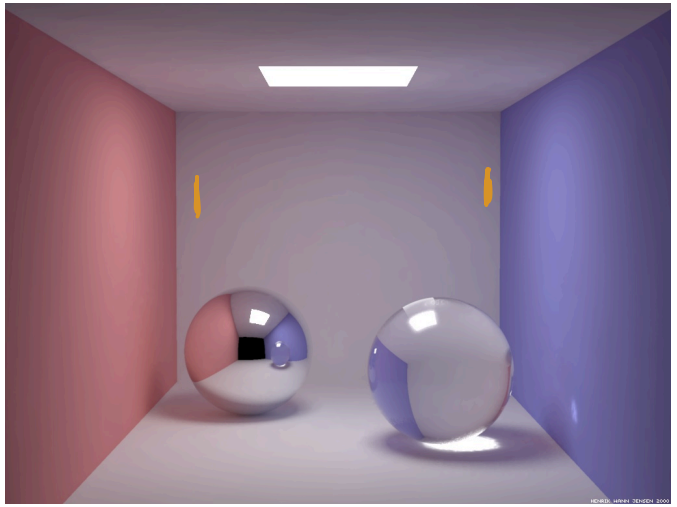
standard ray tracing



+soft shadows



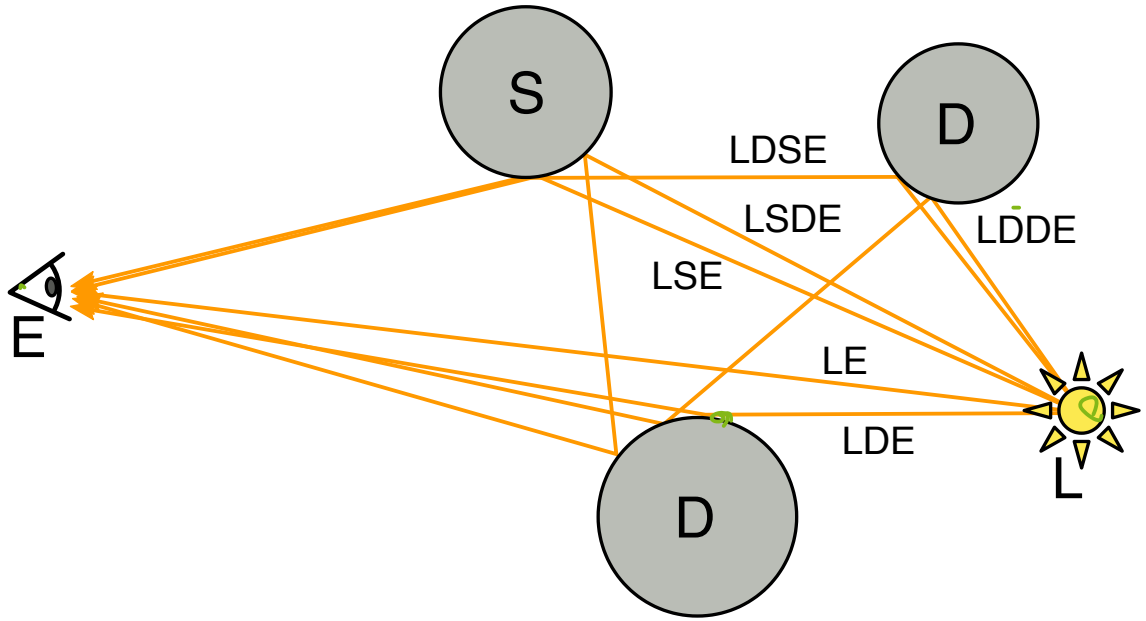
+caustics



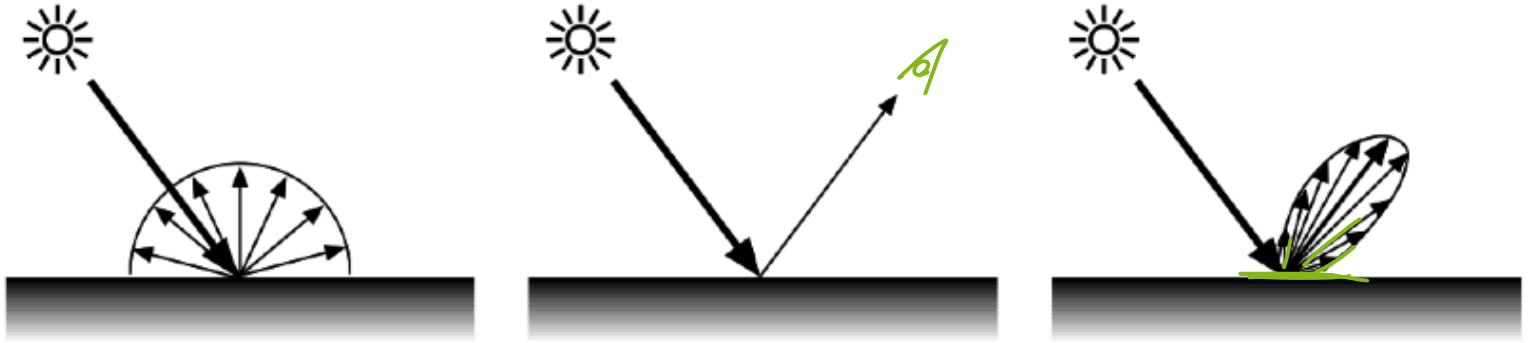
+indirect lighting

Light Paths

- E = eye point
- L = light source
- D = diffuse reflection
- S = specular reflection



Types of Reflections



D: diffuse

S: specular

G: glossy

H. Lensch, “Efficient Image-Based Appearance Acquisition of Real-World Objects”, PhD thesis, 2004

Quiz: Light Paths

- Which light paths can a standard recursive ray-tracer handle?
- Notation:
  - $+$ = one or more occurrences
  - $*$ = zero or more occurrences
  - () = grouping
  - $|$ = or

A: ED^*L

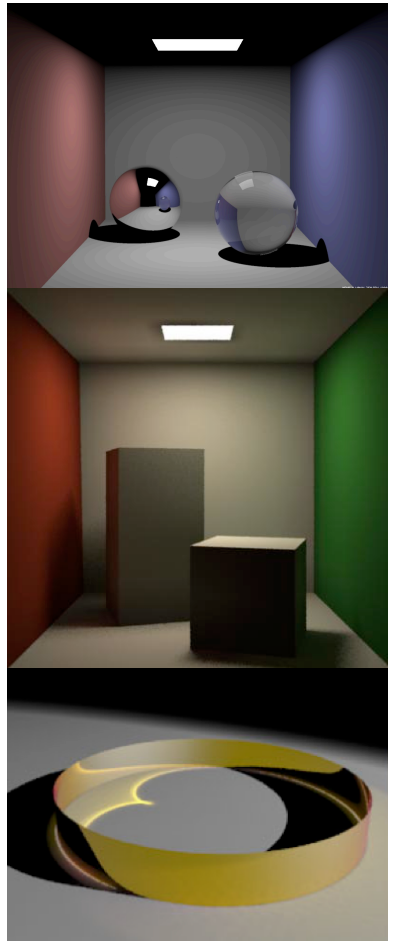
B: $E[(D|G|S)^+(D|G)]L$

C: $E(D|G)[S^*]L$

D: $E[S^*](D|G)L$

Recursive Ray Tracing

- Which light paths can a standard recursive ray-tracer handle?
- Only $E[S^*](D|G)L$



- No multiple diffuse inter-reflections
  - E.g. $EDDL$
- No caustics
  - E.g. $EDSL$